

DRAG AND DROP STATELESS DATA CLASS SPECIFICATION AND PROGRAMMING

Technical Field

The present invention relates to a system, method, and article of manufacturer
5 for programming processing operations within a server and more particularly to a
system, method, and article of manufacturer for programming processing operations
within a multi-tier client-server architecture.

Background

Building scalable client-server based applications has increased in importance
10 recently with the increase in the use of the Internet as a mechanism to connect potential
users of data, data sources, and processing resources with these various data sources and
processing resources. These applications that previously were required to scale to a
potential of 100 or more simultaneous users are now potentially exposed to millions of
simultaneous users of the Internet. To effectively build these applications, multi-tiered
15 application architectures that are implemented using a multi-tiered processing
architecture have been developed to provide the needed level of scalability.

In prior implementations of these applications, a client manipulating a database
is directly connected to the database and maintains that connection during manipulation
of the data. This type of architecture allows for easy data synchronization between the
20 client and the database. However, this type of computer architecture places limits on
the number of clients that might work with the database at any one time; or in other
words, this architecture is not scalable. In other data structure architectures, a multi-tier

architecture, such as a three-tier architecture, is used. The client comprises the first tier. The second tier includes servers that contain the business logic that may be easily replicated or scaled to meet the needs of the number of simultaneous users of the system. The third tier contains the one or more database connected to the second tier
5 processors.

In the multi-tier architecture, the client makes a request to the second tier for data. A server in the second tier pulls the data from the database located within the third tier and sends a snapshot of the data to the client. The client can make changes to the data, such as update the data, add data, or delete data. The changes are sent back to
10 the second tier. Upon return of the data from the client, the client request may or may not be routed to the same server that sent the data to the client originally. Each processing request from the client to the server/database is separately routed to an appropriate server within the second tier of the architecture based upon any number of factors related to the current processing loads and operating environments of the servers
15 when the request is made.

Because of the fact that data sent between a client and a database may be routed through different processing servers, a stateless organization of data objects is needed. In such objects, each processing operation does not depend upon the processing module performing an operation to know what has occurred in prior operations that may have
20 been performed. As a result, each request may be handled by any server on a request by request basis.

Programming the interaction of the various levels of a multi-tiered architecture

for these applications using a stateless data object design presents a number of programming problems that ensure that each level of the architecture is operating upon a consistent implementation of the data model. The present invention provides a method, system, and article of manufacturer for constructing a consistent
5 implementation of a stateless data object that is implemented across all of the processing levels in a multi-tiered processing architecture.

Summary

In one aspect of the present invention, a method of tracking data through a multi-tier computing architecture is provided. The method includes dragging a
10 graphical representation for a server processing resource from a server explorer module to a visual design surface module to add a processing to a programmable data object being created in the visual design surface module, identifying data schema associated with the server processing resource added to the programmable data object, creating a typed dataset containing the data structures corresponding to the data schema associated
15 with the server processing resource, creating a command adapter to provide data transfer commands within the programmable data object between the programmable data object and the server processing resource, and creating a data transfer connection between the programmable data object and the server processing resource.

In another aspect of the present invention, a computer program product readable
20 by a computing system and encoding instructions for a computer process for tracking data through a multi-tier computing architecture is provided. The computer process comprises instructions analogous to that described above.

In another aspect of the present invention, a computing system creating programmable data objects for use in a multi-tier computing architecture is provided. The computing system includes a memory module, a user interface module, a mass storage system, and a programmable processing module. The programmable processing module performing a sequence of operations used to implement a computing process analogous to the method described above.

In another aspect of the present invention, a system for creating programmable data objects for use in a multi-tier computing architecture. The system has a server explorer module for presenting one or more processing resources present on a server to a programmer for use in creating a programming object class and a visual design surface module for performing the operations associated with creating, editing, and saving the programming object. The visual design surface module includes a drag/drop module for enabling a programmer to select a server resource from the server explorer module and place the selected server resource within a data object on the visual design surface module. The visual design surface module also includes a command adapter function generation module for generating a data processing object associated with the drag and drop of a server processing resource, a typed dataset generation module for generating typed dataset object associated with the drag and drop of a server processing resource, an init generation module for generating the set of data processing functions and methods associated with the drag and drop of a server processing resource, and a properties edit module for retrieving the properties and source code for the various objects within the visual design surface module for editing.

The invention may be implemented as a computer process, a computing system, or as an article of manufacture such as a computer program product. The computer program product may be a computer storage medium readable by a computer system and encoding a computer program of instructions for executing a computer process.

- 5 The computer program product may also be a propagated signal on a carrier readable by a computing system and encoding a computer program of instructions for executing a computer process.

A more complete appreciation of the present invention and its scope may be obtained from the accompanying drawings, that are described below, from the
10 following detailed descriptions of presently preferred embodiments of the invention and from the appended claims.

Brief Description of the Drawings

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

- 15 Fig. 1 illustrates an Internet based web server system utilizing an improved database system according to one embodiment of the present invention.

Fig. 2 illustrates a multi-tier architecture that may be used to implement aspects of an embodiment of the present invention.

- 20 Fig. 3 illustrates a computing system that may be used to implement aspects of the embodiments of the present invention.

Fig. 4 illustrates an example embodiment of a component designer module used to create component datasets and supporting programming object classes according to the present invention.

Fig. 5 illustrates another embodiment of a component designer module used to create component datasets and supporting programming object classes according to the present invention.

Fig. 6 illustrates an example embodiment of a virtual design surface module used to create component datasets and supporting programming object classes according to the present invention.

Fig. 7a and 7b illustrate an example set of programming instructions used to implement FillDataSet and UpdateDataSet functions according to one embodiment of the present invention.

Fig. 8a and 8b illustrate an example set of programming instructions used to implement FillAllDataSet and UpdateAllDataSet functions according to one embodiment of the present invention.

Fig. 9 illustrates an example class programming object created by a component designer module according to one embodiment of the present invention.

Fig. 10 illustrates an operational flow diagram for creating an example class programming object created by a component designer module according to one embodiment of the present invention.

Detailed Description

In the following description of preferred embodiments of the present invention, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be
5 practiced. It is understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

In general, the present disclosure describes a system, method, and article of manufacturer for programming processing operations within a multi-tier client-server architecture. The multi-tier client-server architecture uses a state-less data transfer
10 programming model in which the remote client communicates with data sources and processing resources present on servers using a sequence of message transfers. These messages are considered stateless when both the client and the server do not use a connected processing model in which both processes immediately communicate changes made to the shared data values as they are being modified or updated. The
15 stateless programming model is a non-connected model in which the processes communicate using messages-response pairs that are treated as separate and independent data transactions.

The stateless programming model is useful in allowing a web server to easily scale using a collection of identical processing devices as each server process within the
20 scaled web server will receive a request message and will in turn generate a response message. Because the messages are able processed as independent transactions, each message-response pair sent and received from a particular client may be processed by a

different server process running on any of the identical processing devices within the scaled web server.

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237

to send a URL which includes request information across the Internet 101 to the web server 110. The request information included within the URL typically specifies a database request. The web server 110 processes the URL to obtain the database request information to be supplied to the database 111 resulting in the invocation of the
5 database request specified by the user.

Once the database request is complete, the web server 110 generates the HTML representation for a web page that contains the data corresponding to a result set generated when the database request is applied to the database 111. This HTML representation of the web page is transmitted back across the Internet 101 to the user's
10 computer 103 for display to a user using the web browser. This process of sending a database request, generating the results, generating the HTML web page representation of the results, and returning the representation to the user occurs each time a user using, his or her computer 103, communicates over the Internet to a web server 110 having a database 111.

15 Because the number of simultaneous requests for data sent to the web server 110 from various client computers 103 can both be large and significantly different for each use of a web server, the architecture of the web server system 100 typically attempts to be scalable in allowing a plurality of identical web servers 110 are interconnected to both the Internet 101 and the database 111. Programming the functionality of the web
20 server system 100 in which such a multi-tier architecture is used requires a programming system 131 that creates and modifies software that is executed within a client computer 103, a web server 110, and a database 111. In addition, the interaction of the operation of the software from each of these levels within the multi-tier

architecture needs to be specified to interact with the other levels of the architecture in a consistent manner. The programming system 131 disclosed herein assists in this programming of a dataset used in the stateless programming model using a dataset designer module 132. The programming system 131 also assists in the creation and
5 modification of the command set executed upon the web server 110 as this communication occurs.

In one embodiment, the web server provides a web service that allows a process executing upon a client computer 102-104 to execute an operation upon the web server by transmission of an http message across the Internet 101. The http message
10 references a function exposed within a class resident upon the web server 110. Data to be processed by the processing operation may be sent as part of the http request. This data may be included within a structured data organization if the data is specified and transmitted within an extensible programming data object such as an XML data object. The data object is processed by the exposed function to generate a response. This
15 response may, itself, include a representation of a resultant dataset that may also be expressed within an XML data object.

Because the design of the dataset to be used within the client 102-104 and the web server 110 and the creation of the sequence of operating instructions needed by the web server 110 to perform these tasks are difficult to create, a programming system 131
20 is used to assist in the specification of these objects. The use of a single, integrated programming tool allows the tool 131 itself to ensure that the specific details and inter-dependencies between the dataset and its supporting instructions are consistent and operate correctly.

Fig. 2 illustrates a schematic representation of a multi-tier architecture that may be used to implement aspects of an embodiment of the present invention. An exemplary environment 200 for implementing embodiments of the present invention includes a multi-tier architecture comprising a client tier 202, a server tier 204, and a database tier 206. This multi-tier architecture is common, for example, in applications related to the World Wide Web or Internet. The client tier 202, the server tier 204, and the database tier 206 might contain any number of computing systems, for example, a computing system 300 as described in connection with Fig. 3. In the embodiment illustrated in Fig. 2, the client tier 202 includes a plurality of client computing systems 208, 210, 212, 214. Likewise, the server tier 204 includes a plurality of server computing systems 216, 218, 220, 222, 224. The database tier 206 includes a database 226. Of course, the database tier 206 could also contain multiple databases. This exemplary environment is commonly referred to as a three-tier architecture.

In this three-tier architecture, the database tier 206 is the ultimate repository of information or data. The client tier 202 provides the user-interface logic for interfacing with a user of the three-tier architecture. The server tier 204 typically contains the business rules and controls the interface between the client tier 202 and the database tier 206. The client tier 202 typically interacts with the server tier 204, which in turn interacts with the database tier 206 to update the database 226 and retrieve data from the database.

Many users can interface with the server tier 204 simultaneously through a router 228. As stated previously, the client tier 202 might contain several, thousands, or millions of client computing systems 208, 210, 212, 214, for example a web browser or

a rich client. All of these computing systems 208, 210, 212, 214 interact with the router 228 sending requests to the server tier 204 for information and receiving information back from the server tier 204. For example, the client computing system 208 might request a list of books for sale from the server tier 204. The client computing system 208 sends the request to the router 228. The router 228 routes the request to a server computing system, such as the server computing system 216, for example a web server or network server. The router 228 might be a dumb router or a smart router as is commonly known. Preferably, the router 228 routes the client requests based on load. In other words, the router 228 determines which server computing system 216, 218, 220, 222, 224 in the server tier 204 has the most capacity available at the time of the request.

The server computing system 216 receives the request from the client computing system 208, through the router 228, and accesses the database 226 to retrieve the requested information. The database 226 sends the requested information to the server computing system 216. The server computing system 216 processes or formats the information and returns the resulting information to the client computing system 208 through the router 228 for use by the user. As the number of requests to the server tier 204 increases, more server computing systems can be added to the server tier 204. Thus, the multi-tier architecture is scalable.

Fig. 3 illustrates a schematic representation of a computing system that may be used to implement aspects of the embodiments of the present invention. An exemplary environment for implementing embodiments of the present invention includes a general purpose computing device in the form of a computing system 300, including at least one

processing system 302. Such a computing system 300 might be used in the client tier 202, the server tier 204, and/or the database tier 206 of the multi-tier architecture 200 described above.

5 A variety of processing units are available from a variety of manufacturers, for example, Intel or Advanced Micro Devices. The computing system 300 also includes a system memory 304, and a system bus 306 that couples various system components including the system memory 304 to the processing unit 302. The system bus 306 might be any of several types of bus structures including a memory bus, or memory controller; a peripheral bus; and a local bus using any of a variety of bus architectures.

10 Preferably, the system memory 304 includes read only memory (ROM) 308 and random access memory (RAM) 310. A basic input/output system 312 (BIOS), containing the basic routines that help transfer information between elements within the computing system 300, such as during start-up, is typically stored in the ROM 308.

15 Preferably, the computing system 300 further includes a secondary storage device 313, such as a hard disk drive, for reading from and writing to a hard disk (not shown), and a compact flash card 314.

The hard disk drive 313 and compact flash card 314 are connected to the system bus 306 by a hard disk drive interface 320 and a compact flash card interface 322, respectively. The drives and cards and their associated computer-readable media
20 provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the computing system 300.

Although the exemplary environment described herein employs a hard disk

drive 313 and a compact flash card 314, it should be appreciated by those skilled in the art that other types of computer-readable media, capable of storing data, can be used in the exemplary system. Examples of these other types of computer-readable mediums include magnetic cassettes, flash memory cards, digital video disks, Bernoulli
5 cartridges, CD ROMS, DVD ROMS, random access memories (RAMs), read only memories (ROMs), and the like.

A number of program modules may be stored on the hard disk 313, compact flash card 314, ROM 308, or RAM 310, including an operating system 326, one or more application programs 328, other program modules 330, and program data 332. A
10 user may enter commands and information into the computing system 300 through an input device 334. Examples of input devices might include a keyboard, mouse, microphone, joystick, game pad, satellite dish, scanner, and a telephone. These and other input devices are often connected to the processing unit 302 through an interface 340 that is coupled to the system bus 306. These input devices also might be connected
15 by any number of interfaces, such as a parallel port, serial port, game port, or a universal serial bus (USB). A display device 342, such as a monitor, is also connected to the system bus 306 via an interface, such as a video adapter 344. The display device 342 might be internal or external. In addition to the display device 342, computing systems, in general, typically include other peripheral devices (not shown), such as
20 speakers, printers, and palm devices.

When used in a LAN networking environment, the computing system 300 is connected to the local network through a network interface or adapter 352. When used in a WAN networking environment, such as the Internet, the computing system 300

typically includes a modem 354 or other means, such as a direct connection, for establishing communications over the wide area network. The modem 354, which can be internal or external, is connected to the system bus 306 via the interface 340. In a networked environment, program modules depicted relative to the computing system 300, or portions thereof, may be stored in a remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computing systems may be used.

The computing system 300 might also include a recorder 360 connected to the memory 304. The recorder 360 includes a microphone for receiving sound input and is in communication with the memory 304 for buffering and storing the sound input. Preferably, the recorder 360 also includes a record button 361 for activating the microphone and communicating the sound input to the memory 304.

Fig. 4 illustrates an example embodiment of a component designer module used to create component datasets and supporting programmable data object classes according to the present invention. The programming system, or component designer module 131, includes a server explorer module 401 and a component visual design surface module 402. The server explorer module 401 presents one or more processing resources present on a server to a programmer for use in creating a programming object class. These resources include various processing resources available for use by a client to perform any number of processing operations. These resources may include a message queue, a file watching process, an event manager, print server process, a file system, and a database.

In an example embodiment, a database 410 present on the server consists of a number of database tables, including a Customer table 411, an Order table 412, an OrderDetail table 413, and an Other table 414. Various tables within this database 410 are used to construct a Middle Tier Stateless Data Object 403 within the component visual design surface module 402. In the example embodiment shown, three of the four tables, the Customer table 411, the Order table 412, and the OrderDetail table 413, are included within the Middle Tier Stateless Data Object 403 in order to permit a client process to communicate with the database 410 through the Middle Tier Stateless Data Object 403.

The three database tables 411-413 are inserted into the Middle Tier Stateless Data Object 403 when a programmer selects each table using an input pointing device such as a mouse, trackball, track pad, or similar input device to place a cursor upon each table's graphical display element, such as an icon, and highlight the tables. The selected tables 411-413 are dragged from the Server Explorer module 401 to the Component Visual Design Surface module 402 and placed within the Middle Tier Stateless Data Object 403. This drag and drop operation causes a set of items to be created within the Middle Tier Stateless Data Object 403. The created items include a command adapter module 451-453 for each of the tables placed in the Middle Tier Stateless Data Object 403.

The placement of a first table from a database 410 causes the creation of a DB Connection Object 421, its corresponding Managed Resource Component module 422, and a Persistent Data Storage module 423. The DB Connection Object 421 includes a set of processing functions to create and manage a data transfer connection between the

Middle Tier Stateless Data Object 403 and the database 410. This data transfer connection is used by the command adapter modules 451-453 at run time to retrieve and store data within the database 410 using the corresponding database tables 411-413.

The Managed Resource Component 422 provides the DB Connection Object 421 with the necessary address and identification information to establish the data transfer connection. The Managed Resource Component 422 obtains this address and identification information from the Persistent Data Storage module 423. This collection of modules permit the data transfer connection to be established to a different database when the Middle Tier Stateless Data Object 403 is deployed by simply changing the contents of the Persistent Data Storage module 423. The Middle Tier Stateless Data Object 403 is created and tested using a sample database 410 that contains a set of database tables 411-414 identical to a set of database tables found in a production database. When the Middle Tier Stateless Data Object 403 is deployed, the Managed Resource Component 422 will allow the data transfer connection to be easily redirected from the sample database 410 to a production database.

As stated above, the drag and drop operation of the database tables causes a command adapter module 451-453 for each of the tables placed in the Middle Tier Stateless Data Object 403. In the example embodiment shown in Fig. 4, a Cust_Cmd adapter module 451 is added to the Middle Tier Stateless Data Object 403 when the Customer Table 411 is dragged to the Middle Tier Stateless Data Object 403. Similarly, an Order_Cmd adapter module 452 is added to when the Order Table 412 is dragged to the Middle Tier Stateless Data Object 403. Finally, an OrderDetail_Cmd adapter module 453 is added when the OrderDetail Table 413 is dragged to the Middle Tier

Stateless Data Object 403. Each of these three command adapters 451-453 permit the Middle Tier Stateless Data Object 403 to retrieve data from the database 410 and to store data into the database 410 using the corresponding database table 411-413. The drag and drop operation causes the database schema, as defined by the definition of the fields, and their corresponding data types, within the database tables 411-413, to be included within the Middle Tier Stateless Data Object 403.

The drag and drop operation of the database tables 411-413 also creates a typed dataset 440 that also includes this schema and typed table information. In the example embodiment of Fig. 4, this dataset 440 comprises a PurchaseOrder Typed DataSet that has a customer table 441, an Orders table 442, and an OrderDetails table 443. The definition of these three tables corresponds exactly to the definition of the corresponding database table definition 411-413 found in the database 410. In addition, any relationships between the fields within the various tables in the database 410 are maintained within the PurchaseOrder Typed DataSet 440. This typed dataset 440 is the dataset exposed to the client process that communicates with the database 410 through the Middle Tier Stateless Data Object 403. This typed dataset 440 may be specified using an extensible programming language, such as XML, along with the data stored within the dataset when a client process is communicating with the Middle Tier Stateless Data Object 403.

The drag and drop operation of each resource from the Server Explorer module 401 also causes the creation of a set of processing methods within a Code Generated Method module 430. These processing methods perform the necessary processing operations to support the functions performed by the resource added to the Middle Tier

Stateless Data Object 403. These methods include any initialization processing that must occur to support the resource. Each resource will have a default set of processing methods to be included in the Code Generated Method module 430.

5 All of the processing items created on the component visual design surface module 402 within a Middle Tier Stateless Data Object 403 contain the programmable data objects and supporting processing functions and methods that when compiled will create an executable data object that will operate within a middle tier of the multi-tier processing system. The component visual design surface module 402 may present the data object 403 as a graphical display of a collection of elements. The component
10 visual design surface module 402 may also display the source code used to specify the processing items and their respective supporting functions and methods. As such, a programmer may alter the source code for any processing item within the component visual design surface module 402 by displaying the source code and making the alterations. When the changes are saved, the component visual design surface module
15 402 will incorporate these changes into all other processing items as necessary.

For example, a programmer may wish to change the data type of a field within a command adapter module. This change will cause the same change to be made to the corresponding field in the typed dataset 440 as well as any necessary changes within the processing functions and methods 430 that support the use of this database table.

20 Fig. 5 illustrates another embodiment of a component designer module used to create component datasets and supporting programmable data object classes according to the present invention. The component object designer module 501 includes a server

explorer module 401 and a visual design surface module 402. The server explorer module 401 presents one or more processing resources present on a server to a programmer for use in creating a programmable data object class as discussed above. The server explorer module 401 presents graphical representations for these resources to a programmer for use in the drag and drop operations.

The visual design surface module 402 includes a set of processing modules to perform the operations associated with creating, editing, and saving the middle tier data object 403. These processing modules in the visual design surface module 402 include a drag/drop module 502, a command adapter function generation module 503, a typed dataset generation module 504, an init generation module 505, and a properties edit module 506. The drag/drop module 502 performs the operations necessary to permit a programmer to select a server resource from the server explorer module 401 and place it within a programmable data object within the visual design surface module 402. The drag/drop module 502 also causes the other processing modules in the visual design surface module 402 to perform their operations to complete the drag/drop process of a server resource onto the visual design surface module 402.

The command adapter function generation module 503 generates programmable data objects associated with the drag and drop of a server processing resource. In the example embodiment of Fig. 4, the command adapter function generation module 503 generates the three command adapter modules 451-453 and all of the processing functions contained within these processing items. The command adapter function generation module 503 also accepts an updated command adapter module that has been

edited by the properties edit module 506 and generates the updated source code for the functions within the command adapter modules.

The typed dataset generation module 504 generates typed dataset object associated with the drag and drop of a server processing resource. In the example embodiment of Fig. 4, the typed dataset generation module 504 generates the typed dataset module 440 and all of the processing functions contained within this dataset object. The typed dataset generation module 504 also accepts an updated typed dataset module that has been edited by the properties edit module 506 and generates the updated source code for the functions within the typed dataset module.

The init generation module 505 generates the set of data processing functions and methods associated with the drag and drop of a server processing resource. In the example embodiment of Fig. 4, the init generation module 505 generates the processing functions and methods within the Code Generated Method module 430. The init generation module 505 also accepts an updated Code Generated Method module that has been edited by the properties edit module 506 and generates the updated source code for the functions within the command adapter modules.

The properties edit module 506 permits a programmer to retrieve the properties and source code for the various objects within the visual design surface module 402 for editing. When a programmer has completed editing an object from the visual design surface module 402, the properties edit module 506 updates the module using the various object generation modules 503-505 discussed above. The properties edit module 506 also identifies all other objects within the visual design surface module 402

that are affected by the changes made by the update operations. Each of these other objects are also updated using the various object generation modules 503-505 discussed above.

Fig. 6 illustrates an example embodiment of a virtual design surface module used to create component datasets and supporting programming object classes according to the present invention. As discussed above, the visual design surface module 402 includes a drag/drop module 502, a command adapter function generation module 503, a typed dataset generation module 504, an init generation module 505, and a properties edit module 506. The drag/drop module 502 includes an explorer interface module 621 to permit a programmer to select a server resource from the server explorer module 401 and place it within a programmable data object within the visual design surface module 402. The explorer interface module 621 uses a user interface module 622 to perform the visual display and command input operations associated with the drag/drop operation. A class generation module 623 causes the other processing modules in the visual design surface module 402 to perform their operations to complete the drag/drop process of a server resource onto the visual design surface module 402.

The command adapter function generation module 503 generates data processing object associated with the drag and drop of a server processing resource. In the example embodiment of Fig. 4, the command adapter function generation module 503 generates two functions, a GetDataSet function and an UpdateDataSet function. The GetDataSet function is generated by a GetDS module 631 which uses the database table definition to define how the database data is to be obtained and presented.

Similarly, the UpdateDataSet function is generated by a UpdateDS module 632 which uses the database table definition to define how the database data is to be maintained and stored in the database. The GetDS module 631 and the UpdateDS module 632 perform these same operations upon an updated command adapter module that has been
5 edited by the properties edit module 506 and generates the updated source code for the functions within the command adapter modules.

The typed dataset generation module 504 generates typed dataset object using a Table Schema module 641, a Relations module 642, and a Views module 643. In the example embodiment of Fig. 4, the typed dataset generation module 504 generates the
10 Purchase Order typed dataset module 440 and all of the processing functions contained within this programmable dataset object. The Table Schema module 641 generates the table records from the database schema within the programmable dataset object. The Relations module 642 generates the relationship data for the fields within the records within the dataset based upon the corresponding relationship data from the database.
15 The Views module 643 generates the database views data for the records within the dataset based upon the corresponding views data from the database. The Table Schema module 641, the Relations module 642, and the Views module 643 also process an updated typed dataset module that has been edited by the properties edit module 506 and generates the updated source code for the functions within the typed dataset
20 module.

The init generation module 505 generates the set of data processing functions and methods using an Init Function module 651, an InitDataSet module 652, and an InitConnection module 653. In the example embodiment of Fig. 4, the Init Function

module 651 generates the processing functions and methods within the Code Generated Method module 430 associated with the command adapter modules. The InitDataSet module 652 generates the processing functions and methods within the Code Generated Method module 430 associated with the type dataset class. The InitConnection module 5 653 generates the processing functions and methods within the Code Generated Method module 430 associated with the data transfer connection between the Middle Tier Data Object and the database. These three modules 651-653 also accepts an updated Code Generated Method module that has been edited by the properties edit module 506 and generates the updated source code for the functions within the command adapter 10 modules.

The properties edit module 506 permits a programmer to retrieve the properties and source code for the various objects within the visual design surface module 402 for editing using a Properties Retrieval module 661. The properties and source code are presented to the programmer, and edits accepted from the programmer by a User 15 Interface module 662. When a programmer has completed editing an object from the visual design surface module 402, a class update module 663 updates the edited object using the various object generation modules 503-505 discussed above. These three modules 661-663 also identify all other objects within the visual design surface module 402 that are affected by the changes made by the update operations. Each of these other 20 objects are also updated using the various object generation modules 503-505 discussed above.

Fig. 7a and 7b illustrate an example set of programming instructions used to implement FillDataSet and UpdateDataSet functions according to one embodiment of

the present invention. The first of these two processing functions perform the GetDataSet function that obtains the database data needed to fill the dataset and places it within the typed dataset for use by the processing operation. The UpdateDataSet function places the changes made to the typed dataset back into the database. Fig. 7a presents an illustrative set of processing code that implements the FillDataSet operation for a particular command adapter. Similarly, Fig. 7b presents an illustrative set of processing code that implements the UpdateDataSet operation for a particular command adapter. The example code shown in Figs. 7a and 7b is modified simply by replacing the "XXXX" with the particular name for the command adapter.

Fig. 8a and 8b illustrate an example set of programming instructions used to implement FillAllDataSet and UpdateAllDataSet functions according to one embodiment of the present invention. In a preferred embodiment, only a FillAllDataSet function and an UpdateAllDataSet function will be exposed for access by a remote client process. These functions are shown in Figs. 8a and 8b. Fig. 8a provides a FillAllDataSet function 801 that contains an instruction 802 filling the data into the dataset that corresponds to each command adapter present in the programmable data object. Fig. 8b provides a UpdateAllDataSet function 811 that contains an instruction 812 updating the data into the dataset that corresponds to each command adapter present in the programmable data object.

Fig. 9 illustrates an example class programming object created by a component designer module according to the example embodiment of the present invention presented in Fig. 4. The PurchaseOrder typed dataset 900 has three records corresponding to the three database tables 411-413 included in the programmable data

object. In this example the Customer Table Record 901 has at least three fields: a Customer ID 911, a Customer Name 912, and a Customer Address 913. Of course, other fields may be present.

The Order Table Record 903 also has at least three files: a Customer ID 921, an
5 Order ID 922, and an Order Date 923. Of course, other fields may be present. The Customer ID 921 in the Order Table Record 902 corresponds to the Customer ID 911 field in the Customer Table Record 901, and defines one relationship maintained from the database schema.

The OrderDetail Table Record 903 has at least four fields: an Order ID 931, an
10 Item ID 932, an Item Quantity 933, and an Item Description 934. Of course, other fields may be present. As above, the Order ID 922 in the Order Table Record 902 corresponds to the Order ID 931 field in the OrderDetail Table Record 903, and defines another relationship maintained from the database schema.

The public processing functions FillAllDataSet 941 and UpdateAllDataSet 942
15 are contained within the processing source code generated for the Middle Tier Data Object 403. Within the FillAllDataSet 941, a call to each of the Fill DataSet functions for the three command adapters are shown. In the FillDataSet function, the order in which the three adapters are filled is set to be top down as defined by the hierarchical definition of the database schema. As such, Customers are filled first, followed by
20 Orders, and then OrderDetails. This order is needed to ensure the proper relationships between the data records as defined by the schema discussed above.

The UpdateAllDataSet 942, however, operates in a bottom up manner first performing the delete operations 943 and then the insert and update operations 944. This bottom up sequence is also needed to correctly update the database.

Fig. 10 illustrates an operational flow diagram for creating an example
5 programmable data object created by a component designer module according to one embodiment of the present invention. The operational flow starts 1001 and a Drag/Drop module 1011 accepts programmers inputs to select a server resource, such as a database table, and inserts the processing item onto the visual component design surface. An ID Schema module 1012 identifies any database schema information,
10 including a database table definition, a set of relationships between the database table's fields and other database tables, and any database views that are to be preserved in the programmable data object being created.

A Gen Code module 1013 generates the processing instructions to implement the Init, FillDataSet and UpdateDataSet functions needed to support the processing
15 resources or command adapter being added to the data object. This processing instructions may be generated in any desired programming language as long as the sequence of instructions implement the Init, Fill and Update functions.

Next, a Gen DataSet Class module 1014 generates a typed dataset based upon the processing resource being added to the data processing object. If the typed dataset
20 exists, the new data structures are added to the typed dataset with all relationships between the fields within the records present in the typed dataset and the fields within

in order to make all of the edits consistent within all items in the programmable data object.

Test operation 1024 determines, based upon an input from the programmer, if additional processing items are to be edited. If test operation 1024 determines more processing items are to be edited, the processing returns to the Select Edit module 1021 to edit the next processing item. If test operation 1024 determines that no more processing items are to be edited, test operation 1025 determines if any additional processing items are to be added to the programmable data object.

If test operation 1025 determines no additional processing items are to be added, the processing ends 1002. If test operation 1025 determines at least one more processing item is to be added, the processing returns to the Drag/Drop module 1011. The processing continues until all processing items have been added and edited.

The logical operations of the various embodiments illustrated herein are implemented (1) as a sequence of computer implemented steps or program modules running on a computing system and/or (2) as interconnected logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance requirements of the computing system implementing the invention. Accordingly, the logical operations making up the embodiments of the present invention described herein are referred to variously as operations, steps, engines, or modules.

The various embodiments described above are provided by way of illustration only and should not be construed to limit the invention. Those skilled in the art will

Year	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100
1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	